

Week 3 Assignment

Shawn R Smith

CST499 Software Technology Capstone

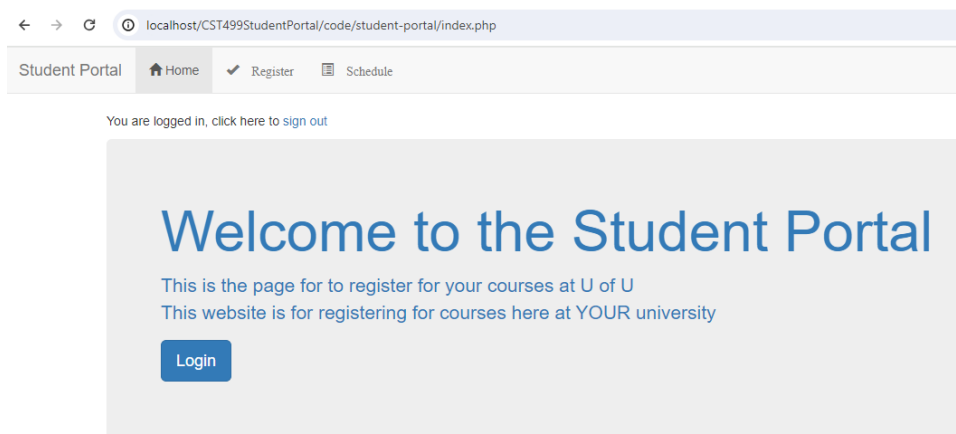
University of Arizona Global Campus

Dr. Butler

February 19, 2024

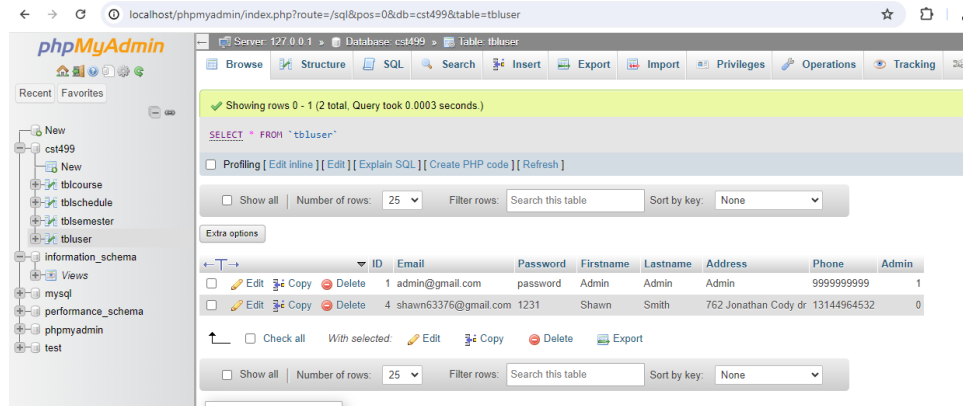
EXHIBITS:

Landing Page



This is the page where users will input their information to be registered in the system.

Data in DB after Registering on Landing Page



This is evidence that the data inputted in the above screen indeed wrote to the DB with the help of the Register.php page behind the scenes.

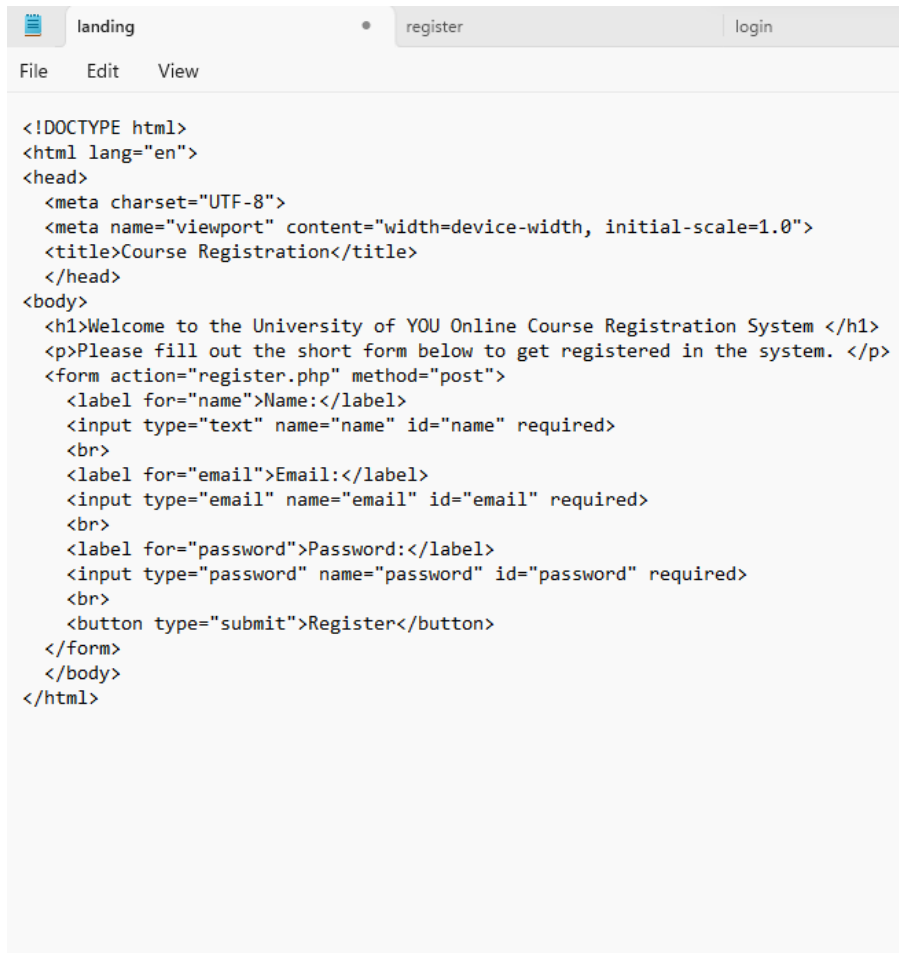
Subsequent Login Page

The screenshot shows the 'Student Portal' login page. It has a navigation bar with 'Home' and 'Register' links. The main content area is titled 'Please Login' and contains two input fields: 'Email address' and 'Password'. Below these fields is a blue 'Login' button. At the bottom, there is a link for 'Not Registered? Click here to sign up'.

This is where users go to login to the system after they have registered.

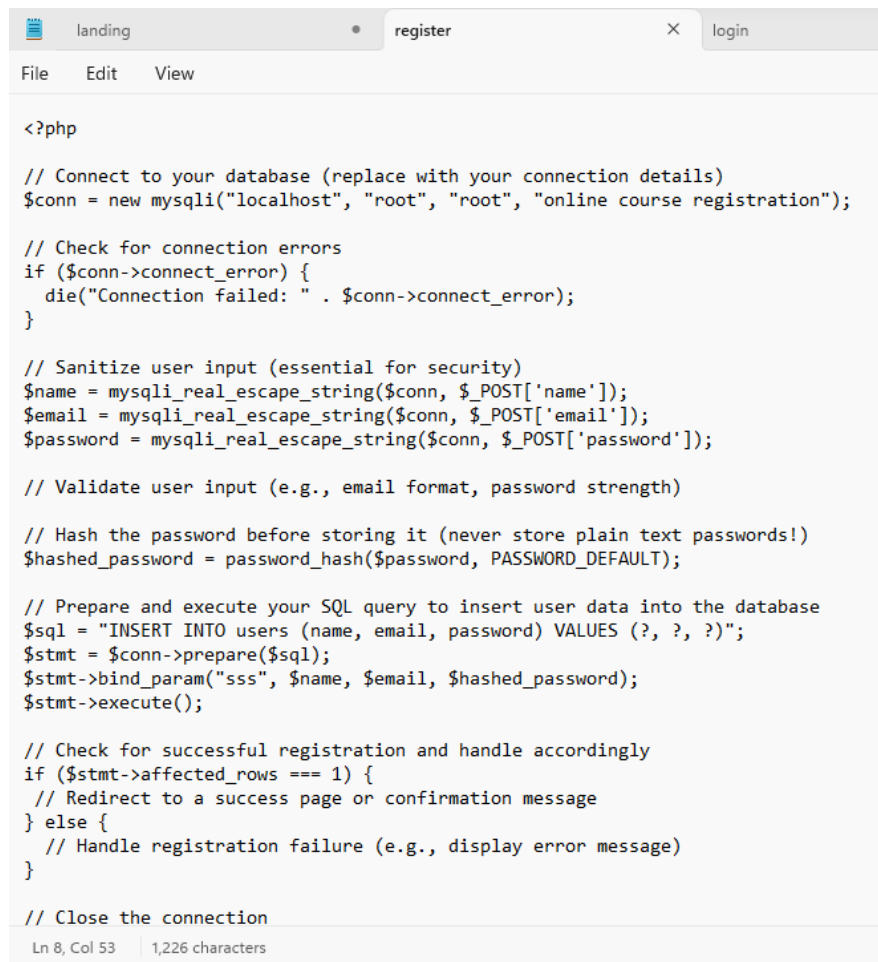
CODE SNIPPETS:

LANDING PAGE CODE

A screenshot of a code editor window. The window has a title bar with three tabs: 'landing' (active), 'register', and 'login'. Below the title bar is a menu bar with 'File', 'Edit', and 'View'. The main area of the editor contains HTML code for a landing page. The code includes a DOCTYPE declaration, a meta charset of UTF-8, a viewport meta tag for mobile devices, a title 'Course Registration', and a body with a welcome message, a paragraph, and a registration form with fields for name, email, and password, and a 'Register' button.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Course Registration</title>
</head>
<body>
  <h1>Welcome to the University of YOU Online Course Registration System </h1>
  <p>Please fill out the short form below to get registered in the system. </p>
  <form action="register.php" method="post">
    <label for="name">Name:</label>
    <input type="text" name="name" id="name" required>
    <br>
    <label for="email">Email:</label>
    <input type="email" name="email" id="email" required>
    <br>
    <label for="password">Password:</label>
    <input type="password" name="password" id="password" required>
    <br>
    <button type="submit">Register</button>
  </form>
</body>
</html>
```

REGISTRATION PAGE CODE



```
<?php

// Connect to your database (replace with your connection details)
$conn = new mysqli("localhost", "root", "root", "online course registration");

// Check for connection errors
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Sanitize user input (essential for security)
$name = mysqli_real_escape_string($conn, $_POST['name']);
$email = mysqli_real_escape_string($conn, $_POST['email']);
$password = mysqli_real_escape_string($conn, $_POST['password']);

// Validate user input (e.g., email format, password strength)

// Hash the password before storing it (never store plain text passwords!)
$hashed_password = password_hash($password, PASSWORD_DEFAULT);

// Prepare and execute your SQL query to insert user data into the database
$sql = "INSERT INTO users (name, email, password) VALUES (?, ?, ?)";
$stmt = $conn->prepare($sql);
$stmt->bind_param("sss", $name, $email, $hashed_password);
$stmt->execute();

// Check for successful registration and handle accordingly
if ($stmt->affected_rows === 1) {
    // Redirect to a success page or confirmation message
} else {
    // Handle registration failure (e.g., display error message)
}

// Close the connection
```

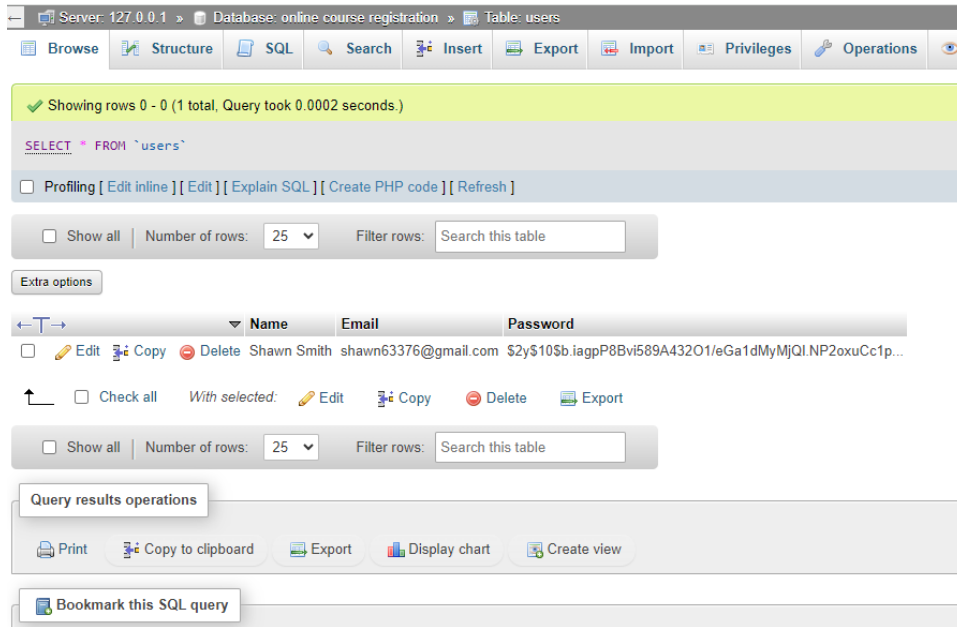
Ln 8, Col 53 | 1,226 characters

LOGIN PAGE CODE

```
landing • register login X
File Edit View

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login</title>
</head>
<body>
  <h1>Login to Access Or Choose Your Courses</h1>
  <form action="login.php" method="post">
    <label for="email">Email:</label>
    <input type="email" name="email" id="email" required>
    <br>
    <label for="password">Password:</label>
    <input type="password" name="password" id="password" required>
    <br>
    <button type="submit">Login</button>
  </form>
  <p><a href="forgot_password.php">Forgot Password?</a></p>
</body>
</html>
```

SQL RESULTS INCLUDING QUERY



1. Explain how to run a PHP file in XAMPP

- Start the XAMPP control panel application.
- Start the Apache and MySQL services.
- Copy the PHP file (written in notepad or another text editor) to the htdocs folder within the XAMPP installation directory.
- Open a web browser and enter the URL <http://localhost/filename.php>, replacing filename.php with the name of your PHP file. (Mikoluk, 2013)

2. Create the landing page, login page, and registration page for new users

- Create three separate HTML files: landing.html (landing page), login.html (login page), and registration.html (registration page).
- Design the structure and layout of each page using HTML elements, such as <header>, <main>, <nav>, <form>, and others.
- Add placeholders for user input fields, such as username, password, email, and other relevant information.

3. Create the MySQL database and tables

- Open the XAMPP control panel and click on the "Admin" button next to the MySQL service.
- In the phpMyAdmin interface, create a new database for the Online Course Registration System.
- Within the database, create a table to store user information, with columns for username, password, email, and any other relevant data.

4. Discuss the MySQL database functions and steps to create the database connection custom class:

- Several MySQL database functions used, such as `mysqli_connect()` to establish a connection to the database, `mysqli_query()` to execute SQL queries, and `mysqli_error()` to handle errors.

5. Develop the registration page layout

- Design the layout of the `registration.html` file using HTML elements, such as `<form>` for the registration form, `<input>` fields for user input, and `<button>` or `<input type="submit">` for the submission button. (Connolly and Hoar, 2018)
- Style the page using CSS to enhance its appearance and make it more aesthetic.

6. Develop the registration page PHP source code

- Create a PHP file (e.g., `register.php`) that will handle the registration process.
- Use PHP to retrieve the form data submitted from the `registration.html` page.
- Validate the user input to ensure it meets the expected criteria (e.g., required fields, email format, password strength).

7. Develop the table that saves the user information in the database

- Write SQL queries or use MyPHPadmin to insert the user information into the database table created above.
- Use the database connection class developed in step 4 to execute the SQL queries and save the user information. (Siteground)

8. Explain the steps taken to create the registration page and save the user information in the database:

- Design the registration form using HTML. This includes input fields for user information like name, email, and password.
- Add form validation using JavaScript (optional) to catch simple errors like empty fields or invalid email formats before submission.
- Include a submit button to trigger the form submission and send data to register.php.
- Establish a secure connection to your MySQL database using a custom connection class and prepared statements.
- Access the submitted data from the \$_POST superglobal variable. (Connolly & Hoar, 2018)

- Implement server-side validation to ensure data integrity and security.
- Never store plain text passwords! Use a secure password hashing function like `password_hash()` before storing it in the database.
- Create a prepared statement to insert user data into the users table.
- Check if the query executed successfully. If so, provide a confirmation message. If not, display an error message indicating the issue.
- Close the DB connection after performing database operations.
- Ensure your database has a users table with appropriate fields for storing user information (name, email, hashed password). Consider additional fields like registration date or user roles.
- Define appropriate data types for each field based on the stored information (e.g., VARCHAR for email, TEXT for long descriptions).

Reference:

Connolly, R., & Hoar, R. (2018). *Fundamentals of web development* (2nd ed.). Pearson.

Mikoluk, K. (2013, September 18). *XAMPP tutorial: How to use XAMPP to run your own web server*

Oh, S. (n.d.). *Bootstrap: Tutorial 7*

SiteGround. (n.d.). *phpMyAdmin create and populate tables tutorial.*

Tsui, F., Karam, O., & Bernal, B. (2018). *Essentials of software engineering* (4th ed.).